VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace integrovaného informačního systému sportovních lokalit obce Třanovice

Design and implementation of the integrated information system for sports facilities of Třanovice municipality

Ostrava 2012 Dalibor Přibyla

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dalibor Přibyla**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Implementace integrovaného informačního systému sportovních lokalit obce Třanovice
Design and Implementation of the Integrated Information System for Sports Facilities of Tranovice Municipality

Zásady pro vypracování:

Cílem této práce je navrhnout a implementovat jednotný informační systém obce Třanovice, který bude spojen s webem obce a bude informovat o sportovních událostech obce. Systém bude rovněž obsahovat rezervační systém, který umožní uživatelům rezervovat si hřiště a ostatní sportoviště. Systém bude navržen tak, aby byl snadno adaptovatelný jako sportovní informační systém jiné obce.

1. Proveďte analýzu a návrh jednotného informačního systému obce.
2. Proveďte rešerši implementační prostředí pro tvorbu webových informačních systémů a nástrojů pro objektově-relační mapování (ORM).
3. Implementujte informační systém ve zvoleném prostředí a s použitím vybraného nástroje pro ORM.
4. Otestujte výkon aplikace pro reálná data, porovnejte výkon zvoleného nástroje pro ORM s ostatními ORM nástroji.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Rostislav Wolný**

Konzultant bakalářské práce: doc. Ing. Michal Krátký, Ph.D.

Datum zadání: 19.11.2010
Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 27. 7. 2012

Dalibor Přibyla

Poděkování

Tímto bych chtěl poděkovat Mgr. Rostislavu Wolnému, vedoucímu bakalářské práce, za všechny podněty, vstřícnost a ochotu při vypracování této bakalářské práce. Dále bych chtěl poděkovat doc. Ing. Michalu Krátkému, Ph.D., konzultantovi bakalářské práce, za cenné rady, věcné připomínky a veškerý čas, který mi při zpracování bakalářské práce věnoval. V poslední řadě chci poděkovat celé mé rodině a přítelkyni za podporu nejen při psaní a programování této práce, ale i během celého studia.

Abstrakt

Předmětem této bakalářská práce je analýza, návrh a implementace informačního systému sportovních lokalit obce Třanovice. První ze čtyř kapitol je zaměřena na implementační prostředí pro vývoj webových informačních systémů. Druhá kapitola obsahuje analýzu počátečních podmínek a požadavků obce a samotný návrh vhodného informačního systému. Třetí kapitola porovnává nástroje objektově-relačního mapování a pro vybrané nástroje předkládá výsledky měření propustnosti dat. Poslední kapitola stručně popisuje technologie využité při implementaci informačního systému navrženého v této práci.

Klíčová slova

Informační systém, Rezervační systém, Analýza systémů, ORM, ASP.NET MVC 3 Framework, Entity Framework, NHibernate.

Abstract

The subject of this bachelor thesis is the analysis, design and implementation of the sports locations information system for the municipality of Třanovice town. The first of four chapters is focused on the web information systems development implementation environment. Second chapter contains the analysis of initial conditions and requirements of the municipality and concrete design of a suitable information system. Third chapter compares object relationship mapping tools and for selected tools gives throughput measurement results. The last chapter briefly describes technologies used during implementation of the information system that was developed as a part of this thesis.

Key words

Information system, Reservation system, System analysis, ORM, ASP.NET MVC 3 Framework, Entity Framework, NHibernate.

Seznam použitých symbolů a zkratek

apod.	-	a podobně
atd.	-	a tak dále
ASP	-	Active Server Pages
API	-	Aplication Programming Interface (rozhraní pro programování aplikací)
č.	-	číslo
EJB	-	Enterprise JavaBeans
GUI	-	Graphical User Interface (grafické uživatelské rozhraní)
HTML	-	HyperText Markup Language
IO	-	Input Output (vstupně výstupní)
IS	-	Informační Systém
J2EE	-	Java 2 Enterprise Edition
JSF	-	JavaServer Faces
JDBC	-	Java Database Connectivity
MVC	-	Model View Controller
Obr.	-	obrázek
ORM	-	objektově-relační mapování
ORM nástroj	-	objektově-relační nástroj
SQL	-	Structured Query Language
SŘBD	-	Systém řízení báze dat
XML	-	Extensible Markup Language

Obsah

1 ÚVOD 1

1.1 Cíle práce 1

1.2 Důvody výběru tohoto tématu 1

1.3 Omezení práce 1

2 J2EE - Java 2 Enterprise Ediiton 2

2.1 Úvod 2

2.2 Systémová architektura 3

2.2.1 Klientská vrstva 3

2.2.2 Webová vrstva 3

2.2.3 Obchodní (Business) vrstva 3

2.2.4 Vrstva - Enterprise information system 4

2.3 Vývojová linie Javy EE 5

3 ASP.NET 6

3.1 ASP.NET WebForms 6

3.2 ASP.NET MVC 6

3.2.1 MVC Architektura 6

3.2.2 Kontrola nad HTML a Http 7

3.2.3 Testovatelnost 7

3.2.4 Postaveno na nejlepších částech ASP.NET Platformy 7

3.3 Shrnutí platformy .NET a Java EE 8

4 ANALÝZA POŽADAVKŮ IS 9

4.1 Proč nový IS? 9

4.2 K čemu má IS sloužit? 9

4.3 Role v IS 9

4.4 Funkce systému 10

5 FUNKČNÍ ANALÝZA 11

5.1 Use CASE model 11

5.2 Případy užití 12

5.2.1 Vyplnění formuláře 12

5.2.2 Provedení rezervace 13

5.2.3 Vytvoření volných termínů 14

6 DATOVÁ ANALÝZA 15

6.1 Vstupy ... 15
6.2 Okolí systému ... 15
6.3 Entity relationship diagram ... 16
6.4 Datový slovník ... 17
6.4.1 Tabulka Uzivatele ... 17
6.4.2 Tabulka Rezervace ... 17
6.4.3 Tabulka Terminy ... 17
6.4.4 Tabulka Lokality ... 17
6.4.5 Tabulka Novinky ... 17
7 OBJEKTOVĚ-RELAČNÍ MAPOVÁNÍ ... 18
7.1 Výhody při použití ORM ... 19
7.2 Nevýhody při použití ORM nástrojů ... 20
8 NÁSTROJE PRO OBJEKTOVĚ-RELAČNÍ MAPOVÁNÍ ... 21
8.1 ADO.NET Entity Framework ... 21
8.1.1 Vrstvy Entity Frameworku ... 22
8.1.2 Mapování pomocí Entity Frameworku ... 22
8.1.3 Schopnosti Entity Frameworku ... 22
8.1.4 Model Entity Frameworku ... 23
8.1.5 Klíčové vlastnosti Entity Frameworku ... 23
8.2 NHibernate ... 24
8.2.1 Mapování pomocí NHibernate ... 24
8.2.2 Model NHibernate ... 24
8.2.3 Klíčové vlastnosti NHibernate ... 25
9 MĚŘENÍ PROPUSTNOSTI DAT ... 26
9.1 Postup měření propustnosti dat ... 26
9.2 Výsledky měření propustnosti dat ... 27
9.2.1 NHibernate ... 27
9.2.2 ADO.NET Entity Framework ... 28
9.2.3 Přístup pomocí SQL ... 30
9.2.4 Porovnání provedených měření ... 31
9.3 Vyhodnocení dosažených výsledků ... 33
10 POUŽITÉ TECHNOLOGIE ... 34
10.1 ASP.NET MVC 3 Framework ... 34
10.2 ASP.NET Web Forms ... 34

10.3SQL Server Express 2008 34
10.4ADO.NET Entity Framework 34
10.5NHibernate 34
10.6JavaScript 34
10.7CSS 35
10.8jQuery a jQuery u.i. 35
Závěr 36
LITERATURA 37
PŘÍLOHY 38

1 ÚVOD

1.1 Cíle práce

Cílem této práce je navrhnout a implementovat jednotný informační systém obce Třanovice s použitím nástroje pro objektově-relační mapování a následně tento nástroj otestovat pro datovou propustnost. Informační systém bude spojen s webem obce a bude informovat o sportovních událostech obce. Systém bude rovněž obsahovat rezervační systém, který umožní uživatelům rezervovat si hřiště a ostatní sportoviště. Systém bude navržen tak, aby byl snadno adaptovatelný jako sportovní informační systém jiné obce.

1.2 Důvody výběru tohoto tématu

Hlavním důvodem proč jsem si vybral toto téma, je samotná problematika informačních systémů, která mě velice zajímá, a chtěl bych se jí věnovat po skončení studia ve své budoucí práci. Dalším faktorem výběru tohoto tématu je to, že jsem si sám mohl zvolit implementační prostředí. Zvolil jsem ASP.NET MVC 3 Framework, což je jeden z nejmodernějším frameworků sloužící pro vývoj webových informačních systémů. Také chci zmínit to, že část mé práce je zaměřena na testování ORM nástrojů pro tento informační sytém, o ORM nástrojích jsem toho moc nevěděl a chtěl jsem se o nich více dozvědět.

1.3 Omezení práce

Třetí kapitola této práce je zaměřena na techniku objektově relačního mapování, dále na ORM nástroje a na měření propustnosti dat u vybraných nástrojů. Vzhledem k velkému množství ORM nástrojů, jejich složitosti a také k mé volbě implementačního prostředí ASP.NET MVC 3 se v této kapitole zabývám detailně pouze vybranými nástroji pro platformu. NET, konkrétně Entity Frameworkem a NHibernate.

Ve své práci jsem se snažil o eliminaci anglických výrazů a pojmů, v některých částech se ale anglické výrazy objevují. Jedná se o výrazy a pojmy, kde neexistuje korektní překlad do českého jazyka a není mým záměrem zmatení čtenáře nepřesným překladem.

Kapitola I. - Analýza prostředí pro tvorbu webových informačních systémů

První kapitola bakalářské práce obsahuje analýzu implementačních prostředí pro tvorbu webových informačních systémů. Vzhledem k rozsahu práce, se nezabývám všemi platformami, které existují, nýbrž jsem se zaměřil na dvě významné prostředí pro vývoj webových informačních systémů, konkrétně na: Java EE a ASP.NET. U ASP.NET se později zaměřuji na oficiální framework ASP.NET MVC 3, který jsem zvolil jako své implementační prostředí pro informační systém. Nakonec uvádím stručné shrnutí těchto platforem. Při popisování platformy Java jsem čerpal především z oficiální dokumentace, tedy z [1] a [2], dále pak ze zdroje [3]. Při popisování ASP.NET jsem čerpal ze svých vlastních znalostí a zkušeností, dále pak z oficiální dokumentace k ASP.NET MVC 3 Frameworku, tedy z [4].

2 J2EE - JAVA 2 ENTERPRISE EDIITON

2.1 Úvod

Architektura Javy 2 Enterprise edition nabízí mnoho voleb jak vyvíjet informační systém na této platformě. Nabízí také mnoho druhů komponentů, jako např. servlety, JSP stránky a jiné. J2EE aplikační servry poskytují mnoho dodatečných služeb. Dobře navržený informační systém by měl být robustní, tedy jeho uživatelé očekávají, že bude spolehlivý a bezchybný. Z tohoto důvodu je důležité rozumět a využívat ty části J2EE, které nám pomůžou vybudovat robustní řešení a musí zajistit to, že píšeme kvalitní kód. Jinými slovy J2EE je sada technologií, metod a doporučení jak provádět vývoj a provozování vícevrstvých aplikací pomocí jazyka Java.

J2EE patří do edice **Java Editions,** tato edice obsahuje následující verze Javy:

- Java 2 Standard Edition (J2SE) - Standardní edice určená pro budování desktopových aplikací a apletů. Tyto aplikace typicky využívá menší počet uživatelů. Jedna se o základ jazyku Java.
- Java 2 Enterprise Edition (J2EE) - Je zaměřena na rozsáhlejší aplikace, které mnohdy využívá spoustu uživatelů najednou, jedná se o nadstavbu nad verzí Standard Edition.
- Java 2 Micro Edition (J2ME) - Tato edice Javy je určena pro aplikace používané především na mobilních zařízeních, jako např. telefony, PDA, pagery a jiné. Také je použita např. v tiskárnách, nebo TV tunerech.

Platforma Enterprise edition je navržená k tomu, aby pomáhala vývojářům tvořit spolehlivé, vícevrstvé a bezpečné síťové aplikace. Podnikové aplikace a informační systémy nejsou užitečné jen pro velké společnosti ale také pro agentury nebo dokonce i vlády. Rysy, které dělají aplikace silné,

jako bezpečnost a spolehlivost, často dělají aplikaci velice složitou. Platforma Javy EE je navržená k tomu, aby redukovala složitost vývoje.

2.2 Systémová architektura

Ve vícevrstvých aplikacích je funkcionalita aplikace rozdělena do izolovaných funkčních oblastí, nazývaných vrstvy. Typická vícevrstvá aplikace obsahuje klientskou vrstvu, střední vrstvu a datovou vrstvu. Prostřednictvím klientské vrstvy uživatel posílá požadavky na střední vrstvu, která následně zpracuje tyto požadavky a ukládá data do datové vrstvy. Java EE se zaměřuje především na střední vrstvu, aby poskytla snadnější, bezpečnější a víc robustní řízení pro podnikové aplikace. [1]

2.2.1 Klientská vrstva

Klientská vrstva se skládá z aplikačních klientů pomoci nichž se přistupuje k Java EE servru, tito klienti jsou obvykle umístěni na jiném stroji než je servr. Klienti posílají žádosti na servr. Servr zpracovává žádosti od klientů a vrací odpovědi zpět ke klientům. Mnoho druhů aplikací mohou býti Java EE klienti, nebývají to často ani Java aplikace. Klienti mohou být samostatné aplikace, internetový prohlížeč, či jiné servry. [1]

2.2.2 Webová vrstva

Webová vrstva se skládá ze součástí, které ovládají interakci mezi klienty a obchodní (business) vrstvou. Slouží především k dynamickému generování obsahu v různých formátech pro klienta. Vykonává základní logiku a uchovává data dočasně v součástech JavaBeans. Také udržuje stav dat pro relaci uživatele. Mezi technologie, které spadají do webové vrstvy patří například:

- Servlety - Java třídy, které dynamicky generují obsah stránek.
- JavaServer Pages (JSP) - Html stránky používající jazyk Java pro generování dynamického obsahu.
- JavaBeans Components - Objekty, které slouží jako dočasné úložiště dat pro stránky aplikace. [1]

2.2.3 Obchodní (Business) vrstva

Obchodní vrstva poskytuje logiku pro aplikace. Je to kód, který poskytuje funkčnost ke specifické obchodní doméně, jako je např. finanční sféra. V dobře navržené aplikaci je jádro funkcionality právě v obchodní sféře. Mezi technologie, které spadají do webové vrstvy, patří například:

- Enterprise JavaBeans (enterprise bean) - Komponenty, které zapouzdřují funkčnost aplikace.
- JAX-WS web servise endpoint - Rozhraní pro vytváření SOAP web services.

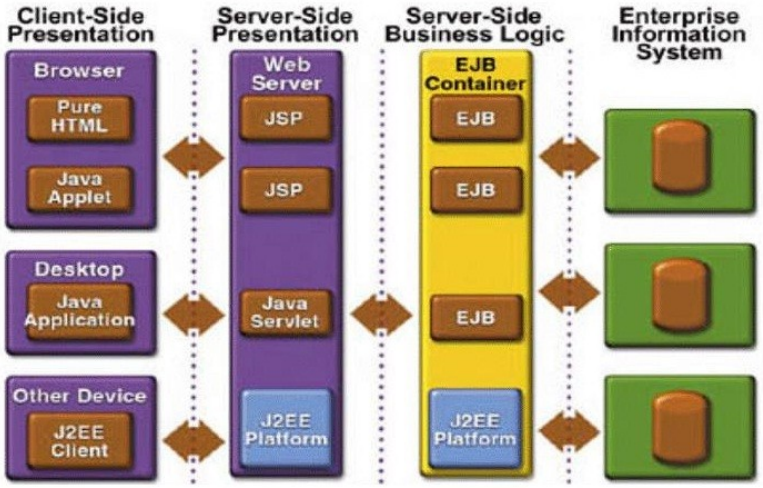
- Java Persistence API entities - Rozhraní pro přístup k datům a mapování na Java objekty [1].

2.2.4 Vrstva - Enterprise information system

Podnikové informační systémy obsahují databázové servry, ty se typicky nachází na jiném stroji než Java EE servr a přistupuje se k nim pomocí komponent z business vrstvy. Technologie, které patří do této vrstvy jsou například tyto:

- The JavaDatabase Connectivity API (JDBC) - Jedná se o rozhraní umožňující přístup k relačním databázím.
- The Java EE Connector Architecture - Rozhraní pro přístup k adaptérům, pomocí nichž se lze připojit k jiným existujícím systémům.
- The Java Transaction API (JTA) - Rozhraní pro definování a správu transakcí.[1]

Pro lepší pochopení těchto vrstev demonstruji rozložení systémové architektury Javy EE na obrázku číslo 2.1



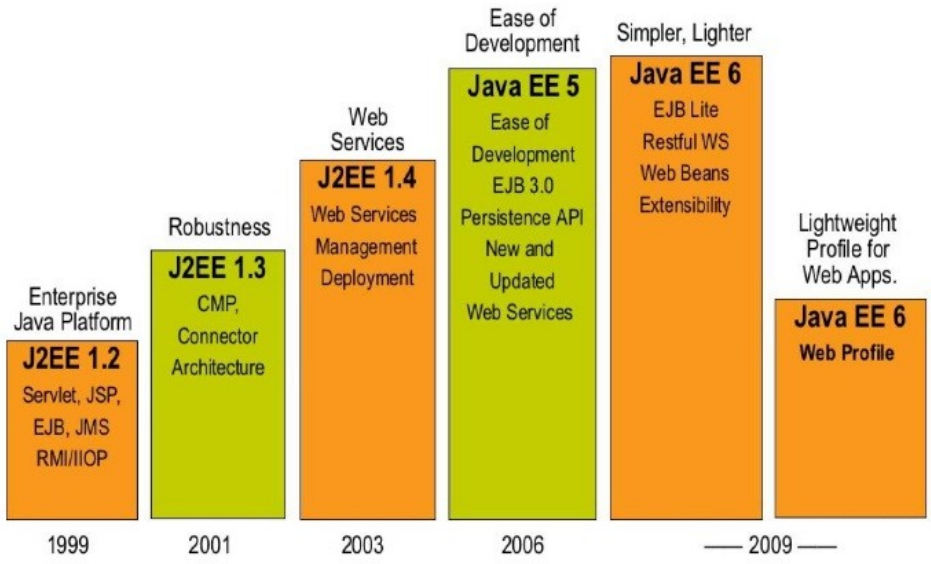
Obr. 2.1 Systémová architektura Javy EE (Obrázek převzat z [2])

Toto rozdělení do vrstev usnadňuje vývojářům programování, aplikace se dají snadněji rozšířit a především také udržovat. Problémy spojené s vývojem informačních systémů se dají rozdělit na menší části, při kterých lze využít různých metodik a návrhových vzorů. Aplikace je také možné integrovat s již existujícími informačními systémy, což je dle mého názorů velice důležité.

Vzhledem k tomu, že jsem nezvolil k implementaci informačního systému obce Třanovice Javu EE tak nebudu zabíhat do detailů této technologie, která je hodně rozsáhlá.

2.3 Vývojová linie Javy EE

Jako poslední bych chtěl uvést, že aktuální verze Javy Enterprise edition je verze 6. Na obrázku číslo 2.2 uvádím časovou linii vývoje této technologie.



Obr. 2.2 Verze Javy EE (Obrázek převzat z [3])

3 ASP.NET

ASP.NET je součást .NET Frameworku, která je určena pro vytváření aplikací a informačních systémů. Technologie ASP.NET je založena na jazyku Common Language Runtime (CLR). Tento jazyk je sdílen všemi aplikacemi, které jsou postavené na tomto vývojovém rámci. Důsledkem toho mohou vývojáři realizovat projekty v kterémkoliv jazyce podporující CLR, např.: C#, C++, Visual Basic.NET a jiné. Aplikace vyvíjené v ASP.NET jsou rychlejší, protože jsou předkompilovány do jednoho, nebo více DLL souborů. To je rozdíl oproti skriptovacím jazykům, kde jsou stránky při každém přístupu znovu parsovány.

3.1 ASP.NET WebForms

Princip WebForms spočívá v tom, že při programování aplikací se využívají ovládací prvky, tzv. Controls. Programování tak velice připomíná vývoj aplikací pro Windows zvaných Windows Forms. To velice ulehčuje přechod k programování webových aplikací zejména uživatelům, kteří mají zkušenosti s programováním Windows Forms. Při tvorbě aplikací je tedy možné používat ovládací prvky jako je například tlačítko (button), popisek (label), zatrhávací rámeček (checkbox), ale i například tabulku ve formě gridu, či jiné ovládací prvky. Tyto ovládací prvky mají své vlastnosti a mohou vyvolávat události. Ovládací prvky vytvářejí HTML kód, který je následně poslán do klientova prohlížeče jako část výsledné stránky.

3.2 ASP.NET MVC

ASP.NET MVC je rámec (framework) určený pro vývoj webových aplikací a informačních systémů od Microsoftu, který kombinuje efektivitu a přehlednost model-view-controller(MVC) architektury, nejnovější myšlenky a techniky agilního vývoje a nejlepší části existující ASP.NET platformy. Je to kompletní alternativa k tradičním ASP.NET Web Forms. Tento vývojový rámec jsem zvolil jako své implementační prostředí pro informační systém Třanovice. Aktuální verze tohoto frameworku (v době psaní této práce) je verze 3.

3.2.1 MVC Architektura

Je důležité rozlišit MVC návrhový vzor a ASP.NET MVC rámec. MVC vzor není nový, sahá do roku 1978 do Smaltalk projektu v Xerox PARC, MVC návrhový vzor získal enormní popularitu a v dnešní době je využíván ve webových aplikacích především z následujících důvodů:

- Uživatelská interakce s MVC aplikací probíhá pomocí „přírodního“ cyklu. Uživatel vytvoří, resp. vyvolá akci a odpovědí aplikace je změna datového modelu a doručení aktualizovaného pohledu uživateli. Následně se cyklus opakuje. Toto je velice výhodné pro webové aplikace, které můžeme chápat jako sérii http požadavků a odpovědí.

- Webové aplikace si vynucují kombinaci několika technologií (např. databáze, HTML a strojový kód), obvykle rozdělených do několika sad či vrstev. Vzory, které vzejdou z těchto kombinací, mapují přirozeně koncepty v MVC.[4]

ASP.NET MVC rámec realizuje MVC vzor, poskytuje velmi dobrou separaci jednotlivých částí aplikace. Ve skutečnosti, ASP.NET MVC realizuje moderní variantu MVC, která se zvláště hodí pro webové aplikace.

3.2.2 Kontrola nad HTML a Http

ASP.NET produkuje čistý, normalizačně vyhovujícího značení. Má v sobě zabudované HTML pomocníky (HTML Helpers), kteří produkují čistý HTML výstup, namísto generování velkého množství HTML, nad kterým má programátor malou kontrolu jak tomu je ve Web Forms. Dobrá synchronizace a podpora s kaskádovými styly (CSS) je samozřejmostí. Z praktických důvodů demonstruji výpisem číslo 3.1 použití několika HTML pomocníků, které jsou použity v informačním systému Třanovice.

```
<div class="editor-label">
    @Html.LabelFor(model => model.Nazev_lokality)
</div>

<div class="editor-field">
    @Html.DropDownListFor(m => m.Nazev_lokality, new SelectList(ViewBag.Lokality))
</div>

<div class="editor-label">
    @Html.LabelFor(model => model.Nazev_novinky)
</div>

<div class="editor-field">
    @Html.EditorFor(model => model.Nazev_novinky)
    @Html.ValidationMessageFor(model => model.Nazev_novinky)
</div>
```

Výpis 3.1 Použití HTML pomocníků

3.2.3 Testovatelnost

MVC architektura umožňuje vyvíjet aplikaci udržitelnou a testovatelnou, protože její části jsou přirozeně odděleny od sebe. ASP.NET MVC 3 má výbornou podporu testovacích jednotek, kde si vývojář může udělat testy na jednotlivé části aplikace a přesvědčit se, že veškeré části aplikace fungují korektně.

3.2.4 Postaveno na nejlepších částech ASP.NET Platformy

Existující platforma od společnosti Microsoft poskytuje osvědčenou sadu součástí a možností jak kvalitně a efektivně vyvíjet webové aplikace. První a nejzřejmější je fakt, že ASP.NET MVC je založeno na .NET platformě, takže umožňuje psát kód v jakémkoliv .NET jazyku a umožňuje přistupovat ke stejným rozhraním, nejenom v MVC samotném, ale také v rozsáhlých .NET knihovnách.

Zadruhé je možno ve frameworku používat rysy jako forms authentication, membership, role, profily a další funkce .NET platformy. Tyto funkce ulehčují vývojáři při vývoji aplikace spoustu věcí a šetří čas. Já jsem například použil speciální rys zvaný membership, který slouží k registrování a správě uživatelů. Jeho výhoda je například v tom, že ukládá do databáze hesla uživatelů zašifrovaná, tím pádem se zvětšuje bezpečnost informačního systému. Membership jsem rozšířil o tabulku uživatelů, kde se ukládají dodatečné informace, toto je doporučený způsob. Z praktického hlediska demonstruji použití tohoto technologického rysu výpisem číslo 3.2, kde je zachycena metoda Register.

```
// POST: /Account/Register
[HttpPost]
public ActionResult Register(RegisterModel model)
{
    if (ModelState.IsValid)
    {
        // Attempt to register the user
        MembershipCreateStatus createStatus;
        Membership.CreateUser(model.UserName, model.Password, model.Email, null, null, true,
        null, out createStatus);

        if (createStatus == MembershipCreateStatus.Success)
        {
            FormsAuthentication.SetAuthCookie(model.UserName, false );

            MembershipUser Prihlaseny_uzivatel = Membership.GetUser(model.UserName);
            string UserID = Prihlaseny_uzivatel.ProviderUserKey.ToString();
            Guid UserID_guid = new Guid(UserID);

            Registrace_do_uzivatelu(UserID_guid, model.Name, model.Surname, model.Telefon);

            return RedirectToAction("Vypis_novinek", "Novinka");
        }
        else
        {
            ModelState.AddModelError("", ErrorCodeToString(createStatus));
        }
    }
    return View(model);
}
```

Výpis 3.2 Registrace nového uživatele pomcí rysu membership

3.3 Shrnutí platformy .NET a Java EE

Platformy .NET a Java Enterprise edition jsou dvě nejrozšířenější platformy určené pro vývoj webových informačních systémů. .NET je platforma společnosti Microsoft. Pro vývoj webových informačních systémů je určená část ASP.NET. U ASP.NET se dále používají rámce Web Pages, Web Forms a MVC, který jsem použil při implementaci já. .NET umožňuje výběr programovacího jazyku, který podporuje CLR. Naopak Java EE je platforma společnosti SUN a je založena na jediném programovacím jazyku, který má stejné jméno, tedy Java. Javu EE můžeme spustit na jakékoliv platformě, tedy na Windows, LINUXu, Solarisu, Macu a jiných UNIXových variantách. Naopak .NET pracuje pouze na jediné platformě a to Windows. Schopnost pracovat na jedné jediné platformě činí spoustu věcí a řešení jednodušší. Důležitý fakt je ten, že obě tyto platformy jsou objektově orientované. Toto je velice významné.

Kapitola II. - Analýza a návrh informačního systému obce Třanovice

Druhá kapitola je zaměřena na analýzu a návrh informačního systému. Analýza je rozdělena do tří částí a to: analýza požadavků, funkční analýza a datová analýza.

4 ANALÝZA POŽADAVKŮ IS

4.1 Proč nový IS?

Lidé v obci Třanovice nejsou informování o sportovních událostech, které se v obci dějí. Navíc nemají možnost využití rezervace sportovišť pomocí internetu, na kterém si mohou pohodlně zarezervovat určitou sportovní lokalitu.

4.2 K čemu má IS sloužit?

Lidé si budou moci pomocí informačního systému zarezervovat školní hřiště, tělocvičnu nebo bowlingovou dráhu Bowling clubu Třanovice. Lidé budou také informováni o sportovních událostech obce, jakou jsou např. různé turnaje a další sportovní akce.

Informační systém bude navržen tak, že v případě vzniku nové sportovní lokality bude možno tuto lokalitu do informačního systému jednoduše přidat.

Dlouhodobým cílem tohoto projektu je zvýšit zájem o sportovní akce a naučit lidi používat internet také ke sportovním aktivitám.

4.3 Role v IS

Tento informační systém budou využívat z hlediska role tito uživatelé:

1) Neregistrovaní uživatelé

Pro neregistrovaného uživatele bude mít tento informační systém pouze informativní charakter a to v podobě čtení novinek v obci, zobrazení kontaktních informací na správce apod. Neregistrovaný uživatel bude moci provést registraci do informačního systému pomocí formuláře, kde vyplní své osobní údaje (jméno a příjmení), kontaktní údaje (e-mail a telefon) a přihlašovací údaje (uživatelské jméno, heslo).

2) Registrovaní uživatelé

Registrovaní uživatelé budou mít rozšířené funkce oproti neregistrovaným uživatelům, a to především v podobě samotných rezervací. Registrovaní uživatelé si budou moci vybrat volný termín na vybrané sportovní lokalitě, který si budou moci zarezervovat. Budou mít také k dispozici výpis svých rezervací, které již dříve provedli v informačním systému. Budou také moci provádět změny ve svém přihlašovacím účtu, a to v podobě změny kontaktních údajů a podobně.

3) Správci sportovních lokalit

Správci sportovních lokalit budou provádět veškeré funkce spojené s rezervacemi. Budou vytvářet, editovat a odebírat volné rezervace. Budou mít také možnost přidat novinku do IS, která se bude vztahovat k dané sportovní lokalitě.

4.4 Funkce systému

V tabulce č. 4.1 uvádím výčet požadovaných funkcí systému. Tabulka je rozdělena do 3 sloupců, 1. sloupec nám říká, o jakou roli uživatele se jedná, v 2. sloupci je uvedena událost, kterou daný aktér vyvolá a v 3. sloupci je reakce systému.

Aktér	Událost	Reakce IS
Neregistrovaný uživatel	Registrace do IS	Zaregistruje uživatele
Neregistrovaný uživatel	Čte novinky a informace z IS	Navigace v IS podle požadavku
Registrovaný uživatel	Přihlášení do IS	Přihlásí uživatele
Registrovaný uživatel	Zobrazení volných termínů	Vypíše volné termíny pro požadovanou lokalitu
Registrovaný uživatel	Provedení rezervace	Zarezervuje termín uživateli a odebere ho z volných termínů
Registrovaný uživatel	Storno rezervace	Odebere rezervaci uživateli a uvolní termín
Registrovaný uživatel	Zobrazení svých rezervací	Vypíše rezervace pro přihlášeného uživatele
Registrovaný uživatel	Zobrazení svých kontaktních údajů	Vypíše kontaktní informace pro přihlášeného uživatele
Registrovaný uživatel	Změna kontaktních údajů	Update údajů pro přihlášeného uživatele
Registrovaný uživatel	Změna přihlašovacího hesla	Změní přihlašovací heslo pro daného uživatele
Správce	Přidání novinky do IS	Zapíše do novinek
Správce	Odebrání novinky z IS	Smaže z novinek
Správce	Editace novinky v IS	Update novinek
Správce	Přidání volných termínů do IS	Vytvoří volné termíny
Správce	Odebrání volných termínů	Smaže z termínů
Správce	Editace volných termínů	Update termínů
Správce	Odebrání rezervace uživateli	Odebere rezervaci, uvolní termín

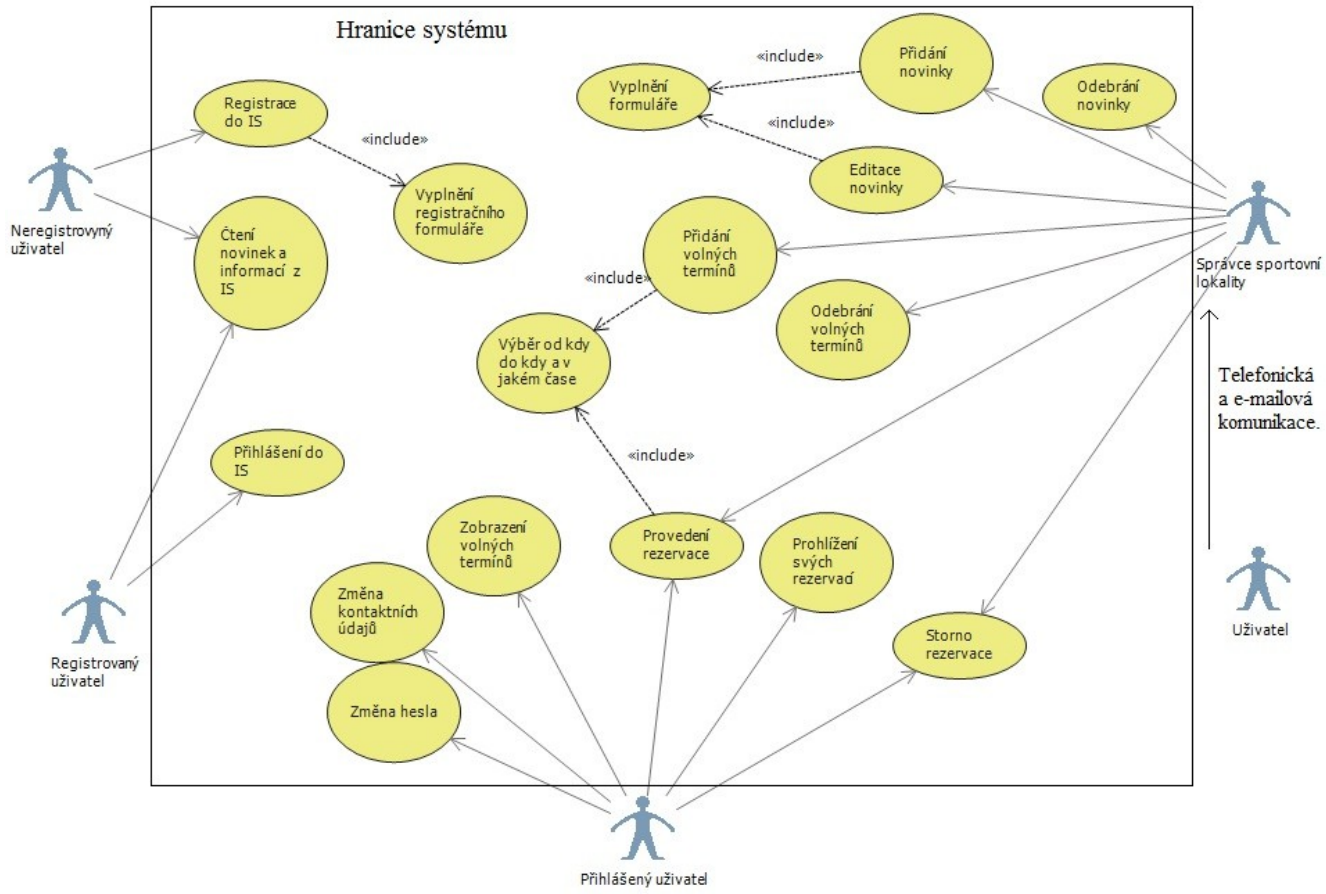
Tabulka 4.1 Požadované funkce systému

5 FUNKČNÍ ANALÝZA

5.1 Use CASE model

V úvodu funkční analýzy jsem vytvořil, Use case model, který je zachycen na obrázku číslo 5.1. Use case model je primárně určen k definici chování systému, aniž by odhaloval jeho vnitřní strukturu. Use case model je základním stavebním prvkem každé analýzy informačního systému a jeho hlavní využití je:

- specifikace požadavků na systém (podklad pro analýzu a návrh),
- komunikace se zákazníkem (uživatelem systému),
- podklad pro řízení projektu,
- tvorba testovacích případů pro fázi testování systému,
- návrh uživatelského rozhraní (mezi každým use case a aktorem by mělo být uživatelské rozhraní).



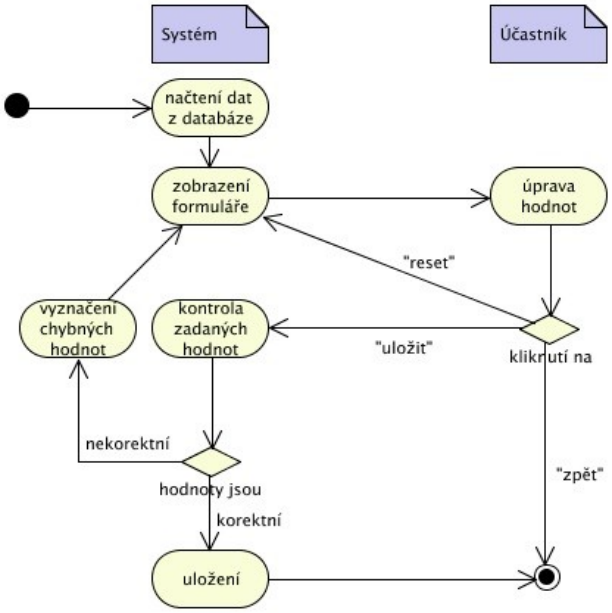
Obr. 5.1 Use case model

Na tomto USE CASE modelu můžeme vidět jednotlivé případy užití pro všechny uživatelské role, hranice systému je vymezené černým obdélníkem.

5.2 Případy užití

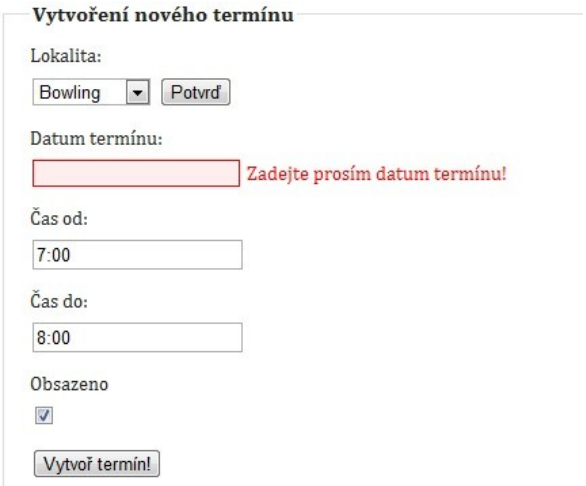
5.2.1 Vyplnění formuláře

Úprava dat v databázi je prováděna pomocí html formulářů. Aby nedocházelo k nekonzistenci dat, je kontrolována před uložením do databáze správnost ukládaných dat. V případě chybně zadaných dat je uživatel informován o tom, jaká data jsou špatně a jaké hodnoty jsou požadovány. K úplnému pochopení uvádím diagram aktivity, který je zachycen na obrázku 5.2. Tento aktivity diagram je obecně platný pro veškeré případy užití týkající se editace databázových položek. Diagram je pomyslně rozdělen na dvě vertikální části, s akcemi systému na jedné straně a uživatele na druhé.



Obr. 5.2 Aktivity diagram pro formuláře

Z praktického hlediska demonstruji použití tohoto kontrolního mechanismu na obrázku 5.3, kde je zachycen špatně vyplněný formulář pro vytvoření jednoho termínu správcem.

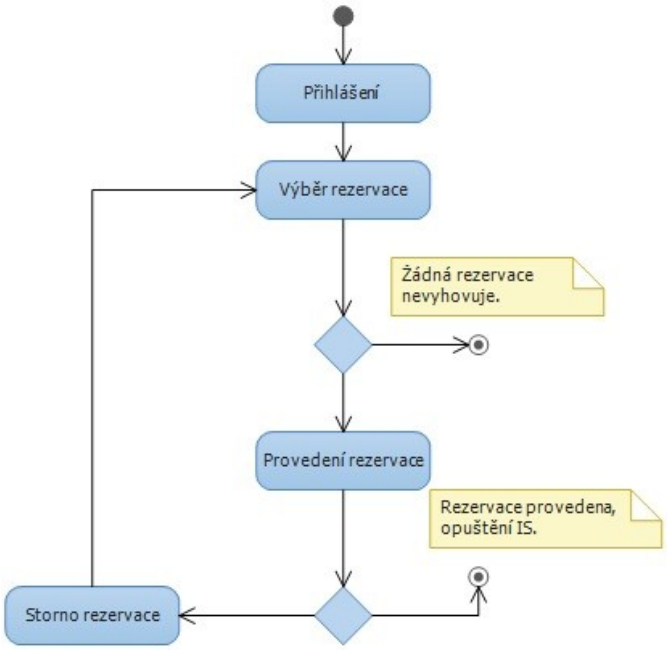


Obr 5.3 Špatně vyplněný formulář

5.2.2 Provedení rezervace

Nejdůležitějšími uživateli informačního systému jsou jeho registrováni uživatelé, protože pro ně v podstatě systém vznikl. Na druhou stranu jsou ale nejvíce omezeni v práci se systémem, a to v tom smyslu, že jsou vlastně pouze externí uživatelé, tj. nemohou používat administrační rozhraní. Proto v této části nezabíhám do podrobností a uvádím jen typický případ užití provedení rezervace.

Pro tento případ užití jsem rovněž vytvořil aktivity diagram, z kterého je vše zřejmé, tento aktivity diagram je zachycen na obrázku číslo 5.4



Obr. 5.4 Aktivity diagram rezervace

Popis užití:

Shrnutí: Provedení rezervace

Účastníci: Registrovaní uživatelé

Vstupní podmínky: Uživatel musí být přihlášen a povolen administrátorem

Hlavní scénář:

1. Uživatel klikne na hlavní straně na odkaz Rezervace.
2. Zobrazí se výběr lokality, uživatel zvolí lokalitu, na které chce provést rezervaci.
3. Z kalendáře si vybere datum.
4. Informační systém zobrazí dostupné termíny a uživatel klikne na odkaz Rezervace. Po tomto kroku je provedena rezervace a uživatel si může své rezervace zobrazit v sekci mé rezervace.

5.2.3 Vytvoření volných termínů

Nejvíce funkcí umožňuje informační systém správci lokality. Správce vytváří volné termíny, novinky, apod. Výčet všech funkcí je zachycen v tabulce 4.1 Zde popisuji složitější případ užití vytvoření volných termínů.

Shrnutí: Vytvoření volných termínů

Účastnící: Správci lokalit

Vstupní podmínky: Uživatel musí být přihlášen v roli správce

Hlavní scénář:

1. Správce klikne na hlavní straně na odkaz Administrátor.
2. Zobrazí se menu, v kterém administrátor klikne na odkaz Vytvoření volných termínů.
3. Správce vybere lokalitu, pro kterou chce vytvořit volné termíny.
4. Správce zvolí datum od kdy, do kdy se mají volné termíny vytvořit.
5. Správce zvolí, v jakém čase se mají termíny vytvořit.
6. Systém vytvoří pro zvolený časový úsek volné termíny, jeden termín je v časovém rozsahu 30min.

6 DATOVÁ ANALÝZA

Informační systém potřebuje ukládat data o registrovaných uživatelích, o tom kdo si zarezervoval danou lokalitu a podobně, proto je nutné provést datovou analýzu a ujasnit si, jaký druh dat je potřeba ukládat a navrhnout strukturu uložení těchto dat.

6.1 Vstupy

a) U **registrovaných uživatelů** bude tento systém evidovat:

ID uživatele - jednoznačný identifikátor pro každého uživatele, jméno uživatele, příjmení uživatele, e-mailovou adresu, telefonní číslo, atribut jestli je uživatel povolen správcem.

b) U **lokalit** bude tento systém evidovat:

Název lokality, město lokality, ulici lokality, PSČ lokality.

c) U **novinek** bude tento systém evidovat:

ID novinky - jednoznačný identifikátor pro každou novinku, název novinky, text novinky, název lokality, pro kterou se daná novinka vztahuje, datum novinky.

d) U **volných termínů** bude tento systém evidovat:

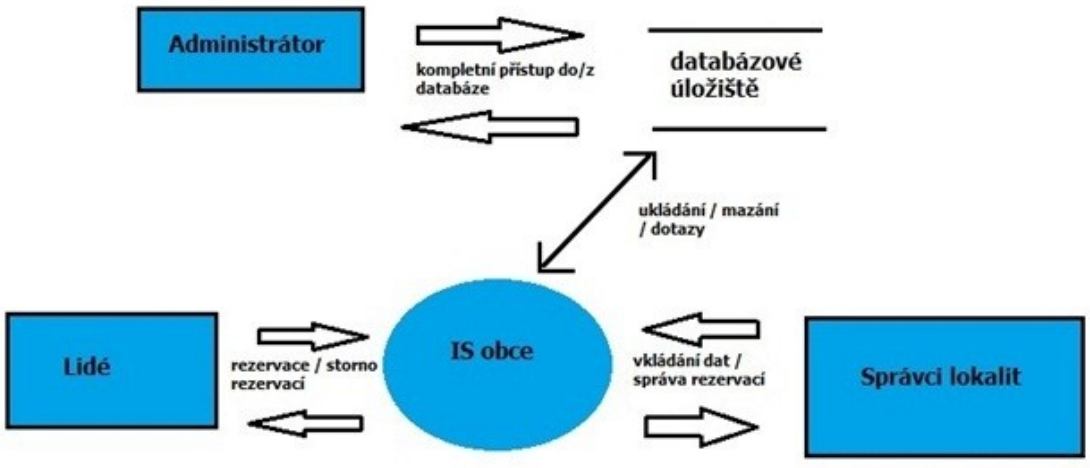
ID termínu - jednoznačný identifikátor pro každý termín, lokalitu, pro kterou se daný termín vztahuje, datum termínu, čas počátku termínu, čas konce termínu, obsazeno - tímto atributem budu rozlišovat jestli je termín obsazen nebo ne.

e) U **rezervací** bude tento systém evidovat:

ID termínu, ID uživatele.

6.2 Okolí systému

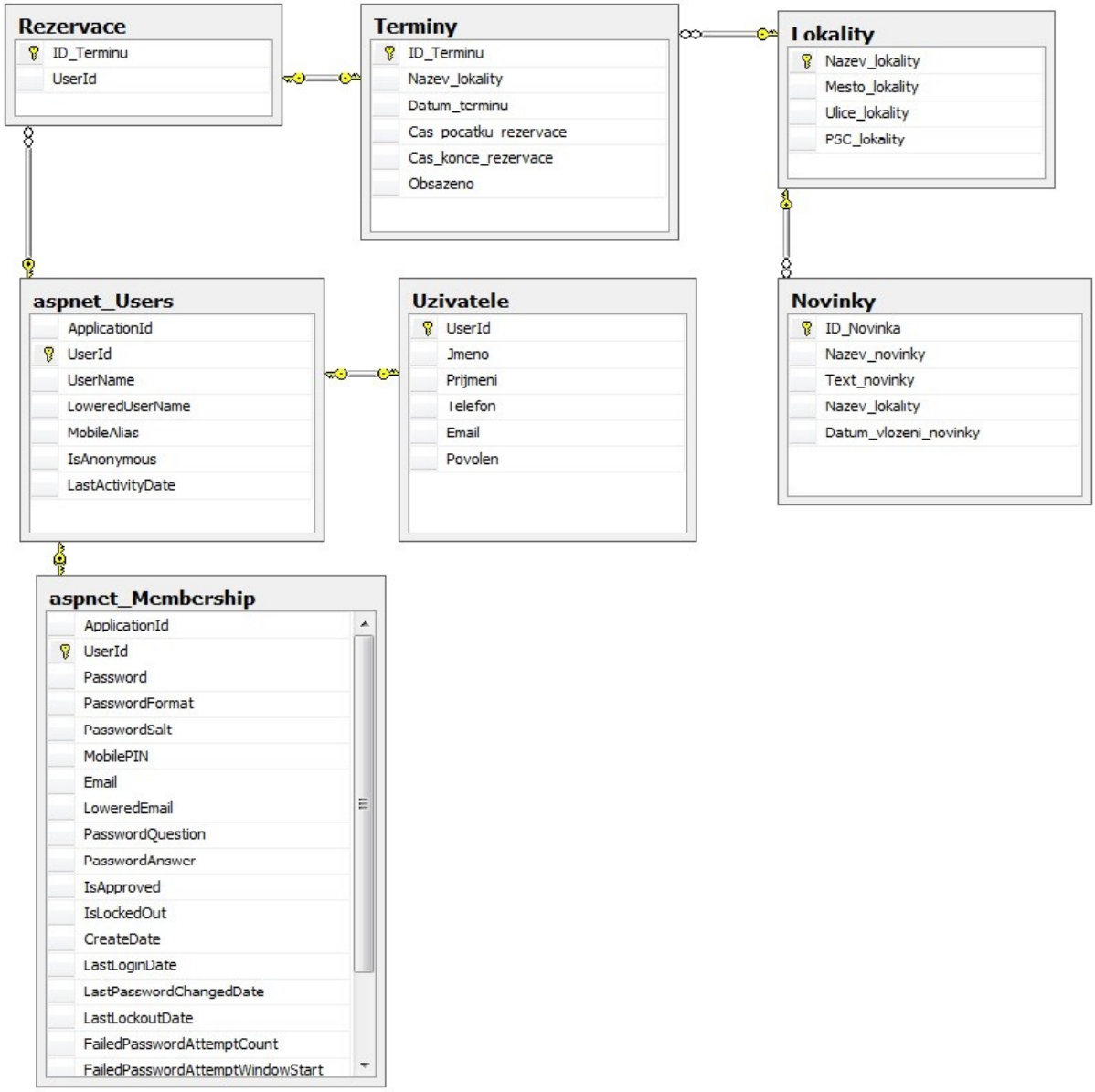
Na obrázku číslo 6.1 ilustruji okolí systému z pohledu datové analýzy



Obr 6.1 Okolí systému

6.3 Entity relationship diagram

Pro znázornění jednotlivých tabulek v databázi jsem vytvořil ER diagram, který je zobrazen na obrázku 6.2.



Obr. 6.2 Entity relationship diagram

V tomto Entity-relationship diagramu vidíme datovou strukturu aplikace, konkrétně jednotlivé tabulky v databázi. Vzhledem k tomu, že jsem si zvolil implementační platformu .NET, systém využívá speciální rys této platformy a to konkrétně Membership pro registrovaní uživatele. Proto pro uživatele jsou evidovány 2 tabulky (aspnet_Users a tabulka Uzivatele). Je to doporučený postup řešení, v případě updatu Membershipu není zasahováno do struktury této funkce.

6.4 Datový slovník

6.4.1 Tabulka Uzivatele

Název	Typ	Velikost	Klíč	Null-able	Index	Popis
ID_uzivatele	uniqueidentifier		pk	ne	ano	jednoznačný identifikátor
Jmeno	varchar	30	ne	ne	ne	
Prijmeni	varchar	30	ne	ne	ne	
Telefon	varchar	20	ne	ne	ne	
Povolen	bit		ne	ne	ne	uživatel musí být povolen správcem

6.4.2 Tabulka Rezervace

Název	Typ	Velikost	Klíč	Null-able	Index	Popis
ID_Terminu	int		pk	ne	ano	cizí klíč z tabulky Termíny
ID_Clovek	uniqueidentifier		fk	ne	ano	cizí klíč z tabulky ASP_NET_ USERS

6.4.3 Tabulka Terminy

Název	Typ	Velikost	Klíč	Null-able	Index	Popis
ID_Terminu	int		pk	ne	ano	jednoznačný identifikátor
Naze_lokality	varchar	50	fk	ne	ano	cizí klíč z tabulky Lokality
Datum_terminu	datetime		ne	ne	ne	
Cas_pocatku_rezervace	varchar	5	ne	ne	ne	
Cas_konce_rezervace	varchar	5	ne	ne	ne	
Obsazeno	bit		ne	ne	ne	

6.4.4 Tabulka Lokality

Název	Typ	Velikost	Klíč	Null-able	Index	Popis
Nazev_lokality	varchar	50	pk	ne	ano	jednoznačný identifikátor
Mesto_lokality	varchar	50	ne	ne	ne	
Ulice_lokality	varchar	50	ne	ne	ne	
PSC_lokality	varchar	6	ne	ne	ne	

6.4.5 Tabulka Novinky

Název	Typ	Velikost	Klíč	Null-able	Index	Popis
ID_Novinka	int		pk	ne	ano	jednoznačný identifikátor
Nazev_novinky	varchar	100	ne	ne	ne	
Text_novinky	varchar	1000	ne	ne	ne	
Nazev_lokality	varchar	50	fk	ne	ano	cizí klíč z tabulky Lokality
Datum_vlozeni_novinky	datetime		ne	ne	ne	

Kapitola III. - Porovnání ORM nástrojů a měření propustnosti dat vybraných ORM nástrojů

Tato kapitola bakalářské práce je zaměřena na techniku objektově-relačního mapování, dále na nástroje pro objektově-relační mapování a na měření propustnosti dat vybraných nástrojů. Úvodem této kapitoly se zabývám samotnou problematikou ORM, pak se detailněji věnuji nástrojům Entity Framework a NHibernate, pro které jsem naimplementoval nástroje určené pro měření propustnosti dat. Při psaní této kapitoly jsem čerpal z poznatků, které jsem nastudoval a zjistil při samotné implementaci ORM nástrojů v rámci informačního systému Třanovice a dále pak z těchto zdrojů [5],[6],[7],[8]

7 OBJEKTOVĚ-RELAČNÍ MAPOVÁNÍ

Objektově relační mapování (ORM nebo O/R mapování) je programovací technika v softwarovém inženýrství, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.[5] Je to tedy technika snažící se automatizovat a zjednodušit přemostění mezi dvěma paradigmaty, a to objektově orientovaným a relačním.[6]

Při modelování aplikací se lidé snaží přirozeně zachytit realitu. Při objektově orientovaném programování jsou ovšem v aplikaci objekty reálného světa reprezentovány jako entity, tzn. instance konkrétní třídy. Zatímco v relační databázi je entita reprezentována jako řádek v databázové tabulce. Rozdílná reprezentace entit v aplikaci a relační databázi je důsledek vzniku programovací techniky, zvané objektově-relační mapování. Díky tomu je programátor odstíněn od psaní SQL dotazů, pomocí nichž se provádí běžné databázové operace jako je čtení, zápis, úprava a mazání dat. Tzv. CRUD operace (create, read, update, delete.) [5]

Hlavním cílem ORM je synchronizace mezi používanými objekty v aplikaci a jejich reprezentací v databázovém systému tak, aby byla zajištěna persistence dat. [5]

Já jsem při implementaci informačního systému zvolil nástroj pro objektově-relační mapování zvaný Entity Framework, díky němu s kombinací s technologií LINQ (Language Integrated Query) jsem nemusel vytvářet žádné SQL dotazy. LINQ je nástroj sloužící pro manipulaci s daty pomocí objektového přístupu, je dostupný od verze 3.5 .NET Frameworku a jazyk C# ho podporuje od verze 3.0. LINQ podporuje dvě metody zápisu:

1) “Compreshension query syntax”
2) “Extension method syntax (Lambda syntax)”

Z praktických důvodů demonstruji výběr pomocí těchto dvou odlišných metod zápisu LINQu na totožném dotazu, který je použit v rámci IS Třanovice výpisem číslo 7.1

```
// Comprehension query syntax
var model = from t in db.Terminy
                where t.Nazev_lokality == Lokalita || (Lokalita == null)
                orderby t.Nazev_lokality, t.Datum_terminu, t.Cas_pocatku_rezervace
                select t;
// Extension method syntax (Lambda syntax)
var model = db.Terminy
       .OrderBy(item => item.Nazev_lokality).ThenBy(item =>
       item.Datum_terminu).ThenBy(item => item.Cas_pocatku_rezervace)
       .Where(item => item.Nazev_lokality == Lokalita || (Lokalita == null));
```

Výpis 7.1 LINQ

7.1 Výhody při použití ORM

Jak už bylo zmíněno, technika objektově-relačního mapování se zpravidla snaží o **úplné odstínění psaní SQL dotazů**, důsledkem toho vzniká nezávislost aplikace na konkrétním databázovém systému, resp. možnost zvolit jiné datové úložiště. Ne všechny ORM nástroje ovšem tuto volbu umožňují. Tuto vlastnost považuji za velice důležitou. Nezávislost na konkrétním databázovém úložišti může být zvlášť přínosná u rozsáhlých projektů, kde by změna znamenala nejen velké finanční náklady na implementaci, ale také by programátoři museli strávit spoustu časů u reimplementace.

V systému IS Třanovice je využit databázový systém Microsoft SQL Server Express, nastavení databázové servru se nachází u ASP.NET MVC 3 Frameworku v souboru Webconfig v části connectionStrings. V parametru providerName nastavujeme, že se bude používat databázový systém Microsoft SQL Server Express.

```
<connectionStrings>
  <add name="Propojeni_EF_s_Databazi"
       connectionString="Data Source=.\SQLEXPRESS;Initial Catalog=IS_Tranovice;
      Integrated Security=SSPI"
       providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Výpis 7.2 Část Web configu

Další výhodou je možnost využití tzv. **líného dotazování** (Lazy loading). Líné dotazování je užitečné, když potřebujeme vykonat více dotazů do několika tabulek databáze najednou. Dochází při něm k řízení dotazů do databáze, které jsou prováděny až v momentě, kdy jsou data spojená s nahrávanou entitou zapotřebí. Dochází tedy při tom k redukci dat, které se musí načíst, a tudíž se nemusí nahrát všechny spojení entit, čímž se rapidně zvýší výkon aplikace.

U nástroje Entity Framework mě velice zaujaly 3 nové techniky, které programátor může, ale také nemusí využít. Jedná se o techniky zvané:

- "**Schema first**" - U této techniky si můžeme pomocí Visual Studia a existující databáze nechat vygenerovat C# třídy pro přístup k databázi.
- "**Model first**" - U této techniky si vytvoříme konceptuální model a z něho lze vygenerovat třídy i novou databázi.
- "**Code first**" - U této techniky si můžeme naopak pomocí tříd napsaných v C# nechat vygenerovat databázi.

Jako poslední výhodu při použití ORM nástroje uvedu to, že programátor může využít některou testovací techniku, která je založena na objektově-orientovaném přístupu.

7.2 Nevýhody při použití ORM nástrojů

Mezi největší nevýhodu ORM nástrojů patří bezesporu snížený výkon aplikací. Některé ORM nástroje vytvářejí plně konceptuální model nad relační databází. To může mít zásadní vliv na výkon aplikace. To lze především vidět v následující kapitole, která se zabývá měřením propustnosti dat u vybraných ORM nástrojů a porovnává je s nízkoúrovňovým SQL přístupem.

Další významný faktor je čas strávený důkladným nastudováním ORM nástroje tak, aby jej programátor mohl využívat se všemi možnostmi, které nabízí. Mně osobně trvalo dost času samotné nastavení a zprovoznění těchto nástrojů. Tento faktor lze ovšem eliminovat zkušenostmi s daným ORM nástrojem. Nutno zmínit, že na internetu jsem nenašel k mnohým ORM nástrojům kvalitní technickou dokumentaci.

8 NÁSTROJE PRO OBJEKTOVĚ-RELAČNÍ MAPOVÁNÍ

V dnešní době existuje velké množství ORM nástrojů, které jsou buďto volně dostupné, tedy využívající freeware licenci, nebo pod placenou licencí. Spoustu firem navíc využívá své vlastní ORM nástroje, ke kterým není vydaná odborná literatura, ani technická dokumentace. V rámci této práce se budu detailněji zabývat nástroji ADO.NET Entity Frameworkem, který je použit v informačním systému Třanovice a jeho konkurentem NHibernate.

Pro platformu .NET existují například tyto ORM nástroje:

- ADO.NET Entity Framework
- NHibernate
- DataObject.Net
- LinqConnect
- BLToolkit
- OpenAccess

Pro platformu Java existují například tyto ORM nástroje:

- Hibernate
- EclipseLink
- iBatis
- Athena Framework
- Ebean
- Object Relational Bridge

Pro obě tyto platformy existuje spoustu dalším ORM nástrojů.

8.1 ADO.NET Entity Framework

ADO.NET Entity Framework je nástroj od společnosti Microsoft a v .NET frameworku se objevuje od verze 3.5 Service Pack1. Enity Framework obsahuje tři vzájemně komunikující vrstvy. Na straně aplikace používá vývojář konceptuální schéma, které je pomocí vrstvy mapování převedeno na schéma databáze. Konceptuální model obsahuje definici asociací, dědění, abstraktních tříd a další objektové vlastnosti. Mapování zde není pouze 1:1, jedna tabulka může být mapována na více tříd. Databáze může pro vývojáře aplikace zůstat stále stejná, i když se fyzicky bude měnit.[4] Implementace ORM Entity Frameworku poskytuje služby, jako je sledování změn, líného dotazování a další, takže vývojáři se mohou zaměřit především na svou specifikaci obchodní logiky a nikoliv na základy pro přístup k datům.

8.1.1 Vrstvy Entity Frameworku

Konceptuální schéma - Objektový model, k této části má přístup vývojář ze strany aplikace

Schéma úložiště (SSDL) - Obsahuje popisy datových tabulek z databáze a jejich datových typů.

Mapovací soubor - Převod CSDL na SSDL

8.1.2 Mapování pomocí Entity Frameworku

Informace o mapování jsou v Entity Frameworku uloženy ve formátu XML, nebo lze přetížit metodu OnModelCreating a mapování nastavit přímo v aplikaci, tento způsob jsem využil já. Přetížení metody demonstruji ve výpisu 8.1

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{

    modelBuilder.Entity<Terminy>().ToTable("Terminy");
    modelBuilder.Entity<Rezervace>()
        .HasKey(p => p.ID_Terminu)
            .Property(p => p.ID_Terminu)
                .HasDatabaseGeneratedOption(DatabaseGeneratedOption.None);

    modelBuilder.Entity<Rezervace>().ToTable("Rezervace");

    modelBuilder.Entity<Lokalita>().ToTable("Lokality");

    modelBuilder.Entity<Novinka>().ToTable("Novinky");

    modelBuilder.Entity<Uzivatele>().ToTable("Uzivatele");

    modelBuilder.Entity<Simple_test>().ToTable("Simple_test");

    modelBuilder.Entity<Simple_test>()
        .HasKey(p => p.Id);
    base.OnModelCreating(modelBuilder);
}
```

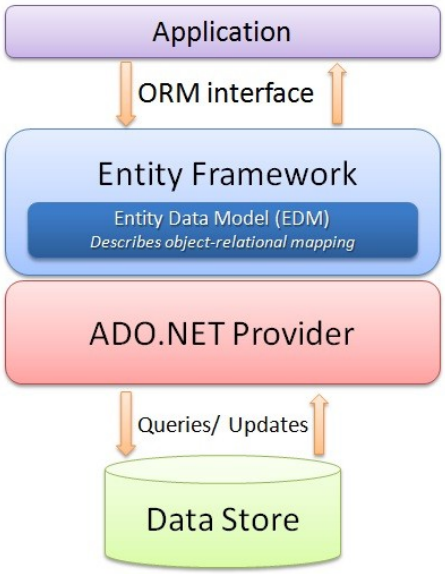
Výpis 8.1 Nastavení mapování pomocí Entity Frameworku

8.1.3 Schopnosti Entity Frameworku

- Umožňuje pracovat s různými databázovými servry (Microsoft SQL Server, Oracle, DB2).
- Obsahuje hodně mapovacích enginu, které zvládají reálné databázové schémata a pracují dobře s uloženými procedurami.
- Poskytuje ingeraci s Visual Studiem.
- Podpora programovací techniky Code first, Model first, Schema first.
- Podpora programovací techniky Code First, která nám umožňuje vytvořit modely entit pomocí kódu, zajímavé je také to, že můžeme mapovat na existující databázi, nebo si vygenerujeme databázi právě z modelu.
- Poskytuje integraci i do všech NET aplikačních programovacích modelů, včetně ASP.NET, Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), and WCF Data Services (formerly ADO.NET Data Services).[8]

8.1.4 Model Entity Frameworku

Entity Framework je postaven na stávajícím modelu poskytovatele ADO.NET, z toho vyplývá, že stávající aplikace postavené na rozhraní ADO.NET mohou být snadno převedeny na Entity Framework. Model je zobrazen na obrázku 8.1.



Obr. 8.1 Entity Framework (Obrázek převzat z [8])

8.1.5 Klíčové vlastnosti Entity Frameworku

- Snížená doba vývoje, rámec poskytuje základní funkce pro přístup k datům, takže vývojáři se mohou soustředit na aplikační logiku.
- Podpora LINQ (Language Integrated Query).
- Mapování mezi objektovým modelem a specifikací úložiště může být změněno bez nutnosti změny kódu.
- M:N mapování.
- Rozsáhlejší dědění pomocí různých přístupů.
- Využitím ADO.NET providerů se lze připojit na libovolnou databázi.[8]

Z uvedených vlastností vyplývá, že Entity Framework je výhodný používat v rozsáhlejších projektech.

8.2 NHibernate

NHibernate je rovněž ORM rámec pro .NET. Pomocí XML popisu entit a vztahů NHibernate automaticky generuje SQL pro načítání a ukládání objektů. Dobrovolně můžeme pomocí NHibernate popsat mapovací metadata s atributy ve zdrojovém kódu. NHibernate podporuje transparentní persistenci, takže třídy nemusí následovat restrikci programovacího modelu. Třídy nemusí implementovat jakákoliv rozhraní nebo dědit ze speciálních tříd. To umožňuje navrhovat byznys logiku používající čisté (CLR) .NET objekty a objektově-orientovaný způsob. NHibernate rozhraní je velice podobné ORM rámci Hibernate, který je určen pro Javu. Veškeré vědomosti a existující dokumentaci k Hibernate je možno používat taky k NHibernate.

8.2.1 Mapování pomocí NHibernate

Mapování je zde realizováno pomocí XML, na výpisu 8.2 demonstruji jednoduché mapování, které jsem použil při měření propustnosti dat tohoto nástroje.

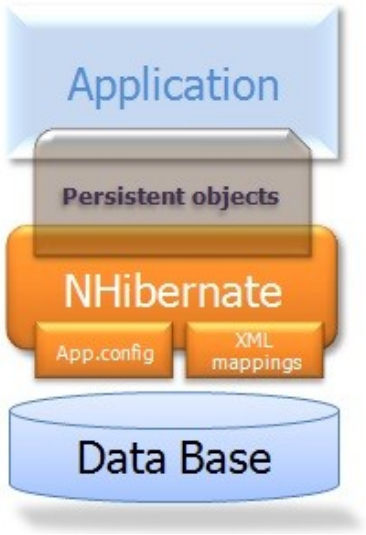
```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
namespace="IS_Tranovice.Domain.Entities"
assembly="IS_Tranovice.Domain">

<class name="Simple_test" table="Simple_test">
<id name="Id" column="Id" type="int">
<generator class="identity"></generator>
</id>

<property name="Value"  column="Value" type="string"/>
</class>
</hibernate-mapping>
```

Výpis 8.2 Ukázka mapování pomocí NHibernate

8.2.2 Model NHibernate



Obr. 8.2 Model NHibernate (Obrázek převzat z [9])

8.2.3 Klíčové vlastnosti NHibernate

- *Přirozený programovací model:* - NHibernate podporuje Objektově-orientovaný způsob, dědičnost, polymorfismus, kompozice, .NET rámce kolekcí, včetně generických kolekcí.
- *Původní .NET* - NHibernate API používá .NET konvenci a způsoby.
- *Negeneruje se žádný build-time bytecode* - NHibernate negeneruje žádný bytecode v průběhu budování procedury.
- *Vlastní SQL* - Nhibernate umožňuje specifikovat konkrétní SQL které má použit k persistenci objektů. Uložené procedury jsou podporovány na Microsoft SQL Servru.
- *Free / open source* - NHibernate je licencováno pod LGPL licencí (Lesser GNU Public License)

9 MĚŘENÍ PROPUSTNOSTI DAT

Za účelem měření propustnosti dat mezi databází a aplikací jsem vytvořil sadu měření a aplikace pro nástroje NHibernate, Entity Framework a nízkoúrovňový SQL přístup. Pro nástroj Entity Framework jsem navíc provedl měření datové propustnosti při dávkovém vkládání termínu rezervací v rámci IS Třanovice.

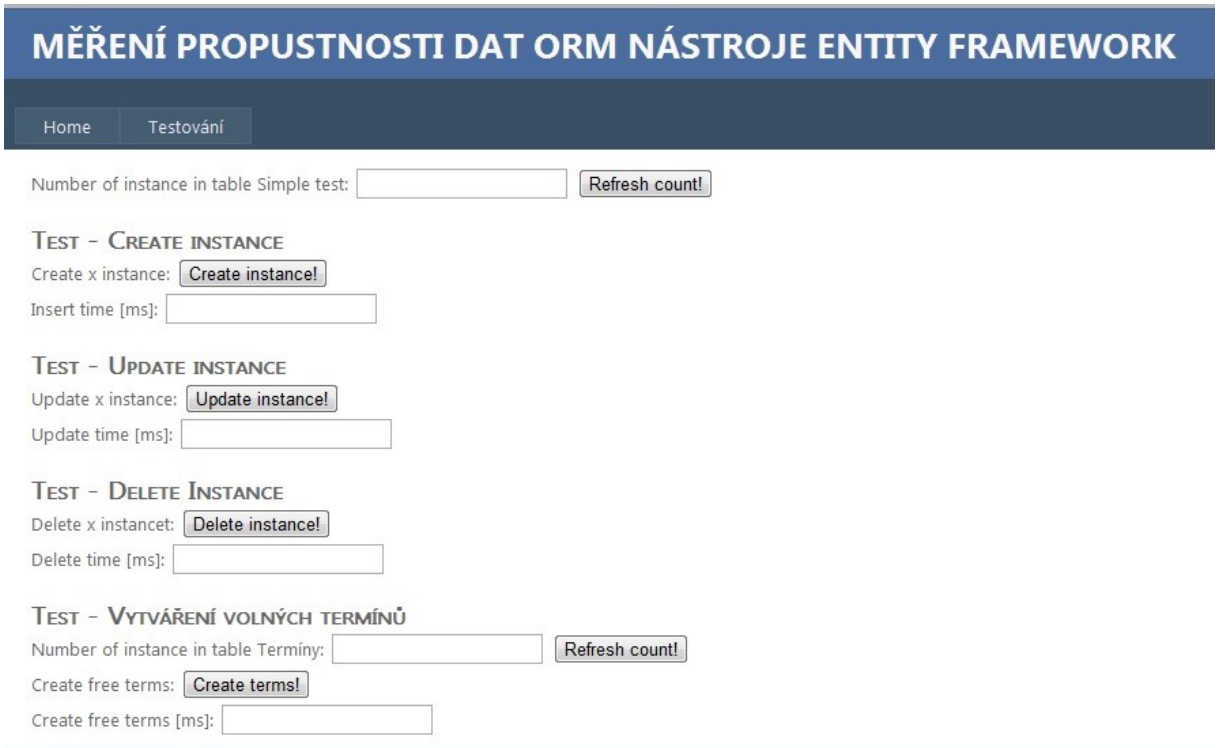
9.1 Postup měření propustnosti dat

1) Vytvořil jsem na databázi jednoduchou testovací tabulku Simple test, kterou demonstruje tabulka 9.1

Název atributu	Datový typ
Id	int
Value	varchar

Tabulka 9.1 Simple test

2) Naimplementoval jsem tři testovací projekty. První využívá ORM nástroj Entity Framework, druhý projekt využívá nástroj NHibernate, třetí využívá nízkoúrovňový SQL přístup. Na obrázku 9.1 demonstruji jeden z těchto měřících nástrojů.



Obr. 9.1 Nástroj pro měření propustnosti dat ORM nástroje ADO.NET Entity Framework

9.2 Výsledky měření propustnosti dat

Pro nástroje Entity Framework, NHibernate i pro přístup pomocí SQL jsem testoval operace vytvoření N instancí, aktualizaci N instancí a smazání N instancí, kde N = 2000, 5000, 10000, 20000.

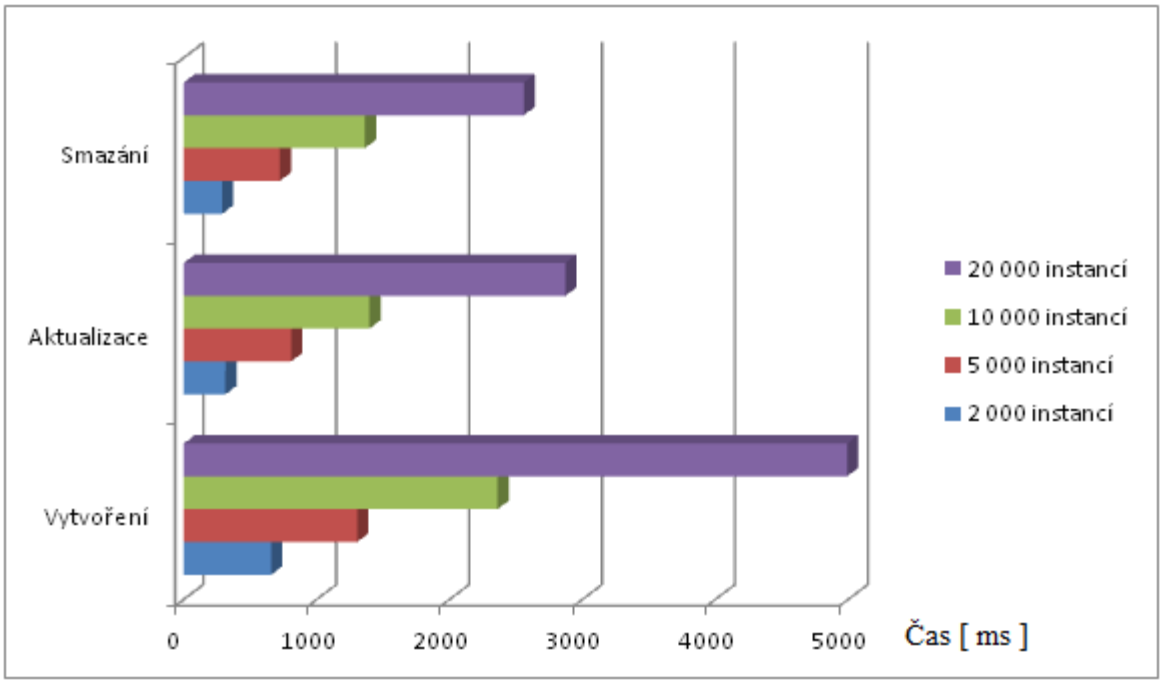
9.2.1 NHibernate

V tabulce číslo 9.2 jsou zaznamenány dosažené výsledky pro měření nástroje NHibernate. Čas uvedený v sekundách je zaokrouhlen na desetiny.

Typ operace	Počet instancí	Čas [ms]	Čas [s]
Vytvoření	2 000	657	0,6
Vytvoření	5 000	1 305	1,3
Vytvoření	10 000	2 362	2,3
Vytvoření	20 000	4 992	4,9
Aktualizace	2 000	314	0,3
Aktualizace	5 000	808	0,8
Aktualizace	10 000	1 397	1,3
Aktualizace	20 000	2 872	2,8
Smazání	2 000	289	0,2
Smazání	5 000	724	0,7
Smazání	10 000	1 364	1,3
Smazání	20 000	2 559	2,5

Tabulka 9.2 Výsledky měření propustnosti dat nástroje NHibernate nad tab. Simple test.

Tyto výsledky jsou dále graficky znázorněny v grafu číslo 9.1. Z grafu je zřejmé, že nejméně časově náročná operace je mazání, následně aktualizace a nejvíce časově náročná operace je vytváření nových instancí.



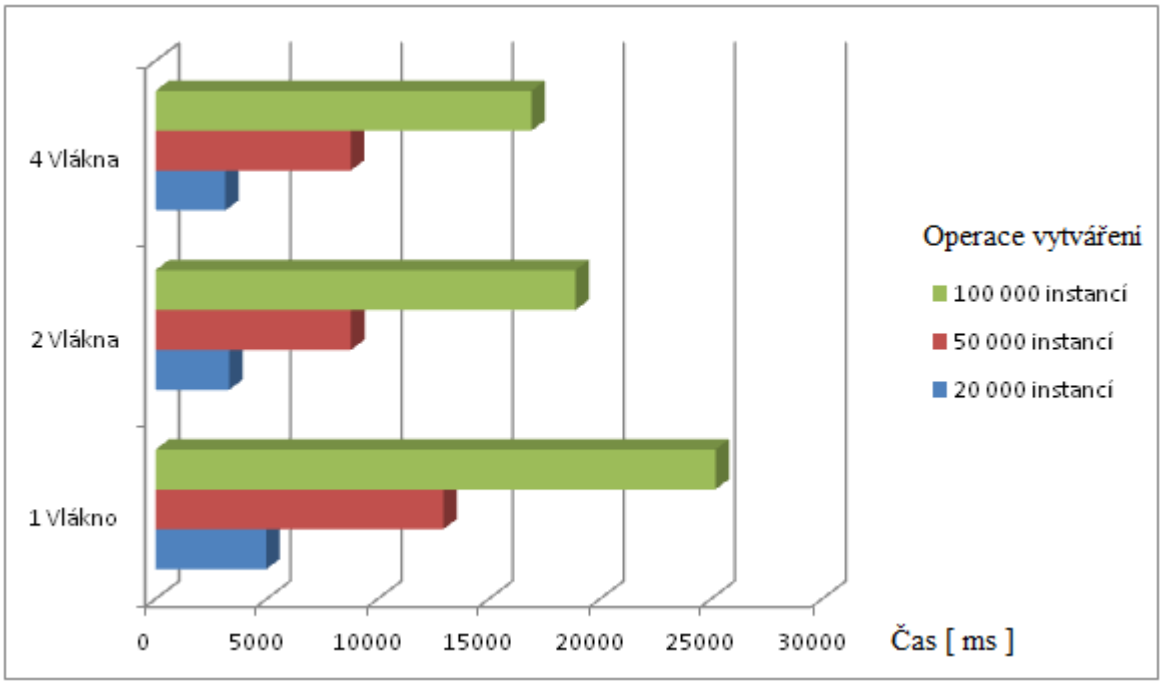
Graf 9.1 Výsledky měření propustnosti dat nástroje NHibernate nad tab. Simple test.

Také jsem pro tento nástroj změřil paralelní přístup pro vytváření nových instancí pro počet N instancí, kde N = 20 000, 50 000, 100 000. V tabulce číslo 9.3 jsou zaznamenány dosažené výsledky pro měření paralelního přístupu při vytváření nových instancí nástroje NHibernate.

Typ operace	Počet instancí	1 Vlákno Čas [ms]	2 Vlákna Čas [ms]	4 Vlákna Čas [ms]
Vytvoření	20 000	4 992	3 324	3 146
Vytvoření	50 000	12 950	8 802	8 791
Vytvoření	100 000	25 201	18 902	16 924

Tabulka 9.3 Paralelní vytváření nových instancí nad tab. Simple test.

Tyto výsledky jsou dále graficky znázorněny v grafu číslo 9.2. Z grafu je zřejmé, že s rostoucím počtem vláken aplikace se zmenšoval čas vložení záznamů, tudíž paralelní přístup byl efektivnější.



Graf 9.2 Paralelní vytváření nových instancí nad tab. Simple test.

9.2.2 ADO.NET Entity Framework

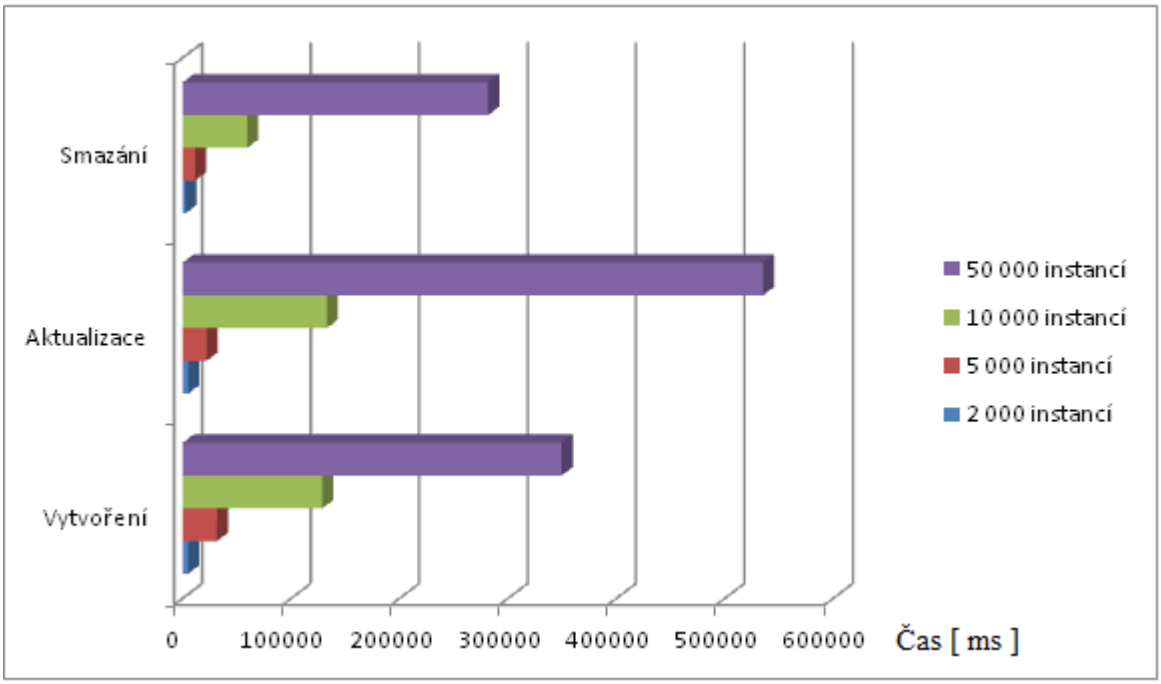
Vzhledem k tomu, že nástroj **Entity Framework** jsem zvolil jako ORM nástroj pro informační systém Třanovice, tak jsem pro tento nástroj neprováděl měření pouze na jednoduché tabulce Simple test, ale měřil jsem operaci využívanou v IS Třanovice, kdy správce vytváří volné termíny. Tato operace je algoritmicky časově náročnější, protože při vkládání dochází ke kontrole duplicit s již vloženými termíny v rámci lokality. Toto měření je pak zobrazeno v tabulce 9.5.

V tabulce 9.4 jsou uvedeny výsledky měření nástroje Entity Framework nad jednoduchou tabulkou Simple test. V této tabulce uvádím i čas v minutách, který je opět zaokrouhlen.

Typ operace	Počet instancí	Čas [ms]	Čas [min]
Vytvoření	2 000	4 968	0,08
Vytvoření	5 000	31 591	0,52
Vytvoření	10 000	128 787	2,14
Vytvoření	20 000	349 578	5,80
Aktualizace	2 000	5 003	0,83
Aktualizace	5 000	22 047	0,36
Aktualizace	10 000	132 934	2,21
Aktualizace	20 000	535 224	8,92
Smazání	2 000	2 607	0,04
Smazání	5 000	11 394	0,18
Smazání	10 000	59 497	0,99
Smazání	20 000	281 655	4,69

Tabulka 9.4 Výsledky měření propustnosti nástroje Entity Framework nad tab. Simple test.

Tyto výsledky jsou dále graficky znázorněny v grafu číslo 9.3. U Entity Frameworku byla časově nejméně náročná opět operace mazání, poté vytváření a nejvíce časově náročná byla operace aktualizace.



Graf 9.3 Výsledky měření propustnosti nástroje Entity Framework nad tab. Simple test.

Jak už jsem se zmiňoval další měření, které jsem prováděl pro nástroj Entity Framework bylo zaměřeno na operaci vytváření volných termínů. Toto měření jsem provedl pro časové úseky jeden měsíc, dva měsíce a tři měsíce. Tento systém je navržen tak, aby byl snadno adaptovatelný i pro více lokalit a proto se vytváří volné termíny. Počítá se s tím, že každá lokalita požaduje vytváření termínů v jiných časových intervalech. Výsledky měření jsou zobrazeny v tabulce číslo 9.5. Čas v sekundách je zaokrouhlen na desetiny.

Časový úsek vytváření termínů	Počet vložených termínů	Čas [ms]	Čas [s]
1 měsíc	1024	2 948	2,94
2 měsíce	1984	4 601	4,60
3 měsíce	2976	13 804	13,80

Tabulka 9.5 Výsledky měření nástroje Entity Framework pro operaci vytváření volných termínů v IS Třanovice.

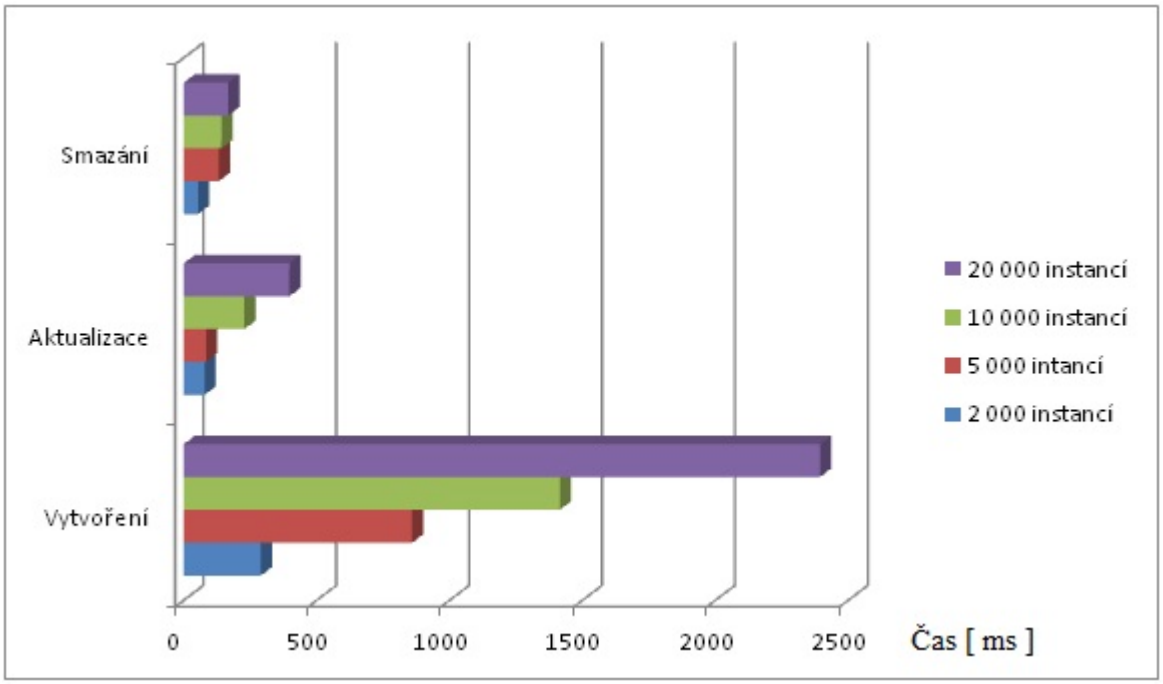
9.2.3 Přístup pomocí SQL

Poslední část měření byla zaměřena na nízkoúrovňový přístup k databázi, tedy pomocí SQL. Dosažené výsledky měření jsou uvedeny v tabulce číslo 9.6. Čas v sekundách je opět zaokrouhlen.

Typ operace	Počet instancí	Čas [ms]	Čas [s]
Vytvoření	2 000	290	0,2
Vytvoření	5 000	859	0,8
Vytvoření	10 000	1 415	1,4
Vytvoření	20 000	2 394	2,3
Aktualizace	2 000	79	0,07
Aktualizace	5 000	84	0,08
Aktualizace	10 000	228	0,2
Aktualizace	20 000	398	0,3
Smazání	2 000	54	0,05
Smazání	5 000	132	0,1
Smazání	10 000	142	0,1
Smazání	20 000	168	0,1

Tabulka 9.6 Výsledky měření pomocí nízkoúrovňového SQL přístupu.

Tyto výsledky jsou dále graficky znázorněny v grafu číslo 9.4. U nízkoúrovňového přístupu byla časově nejméně náročná opět operace mazání, poté aktualizace a nejvíce časově náročná byla operace vytváření nových instancí.



Graf 9.4 Výsledky měření pomocí nízkoúrovňového SQL přístupu.

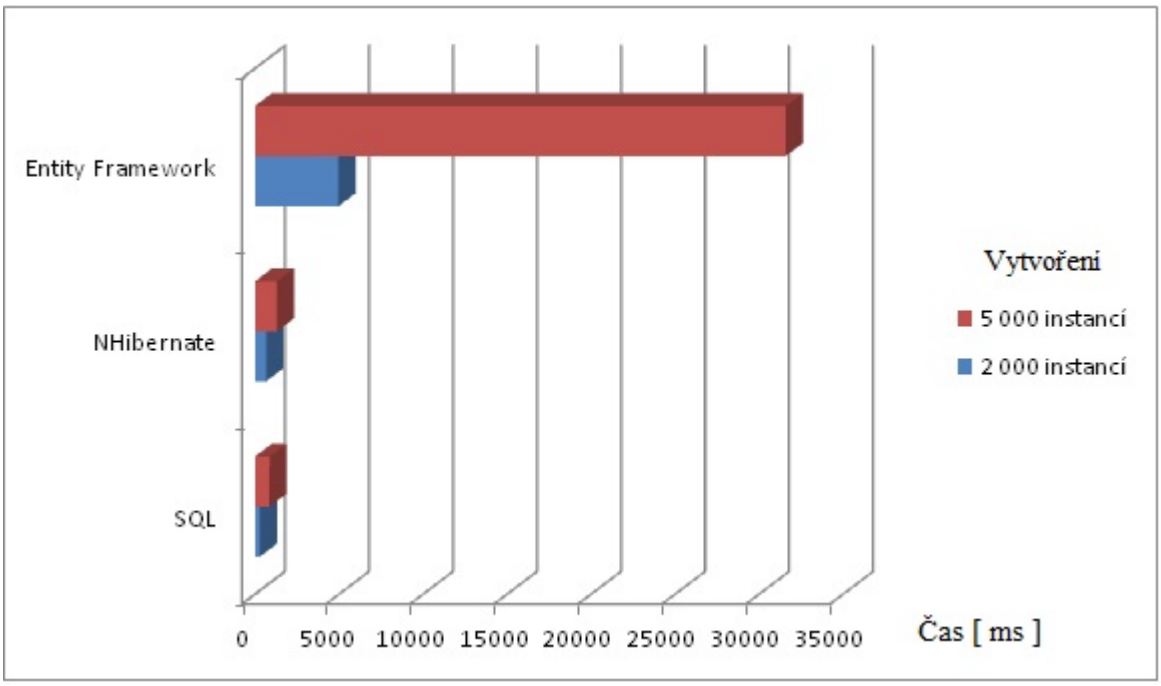
9.2.4 Porovnání provedených měření

Z důvodu souhrnného porovnání těchto tří sad měření jsem vytvořil graf pro každou operaci, kde jdou jasně vidět časové rozdíly mezi nástrojem NHibernate, Entity Frameworkem a přístupem pomocí SQL.

Také chci zmínit, že tyto testy byly prováděny na mém osobním notebooku ASUS F5RL s následnou hardwarovou konfigurací:

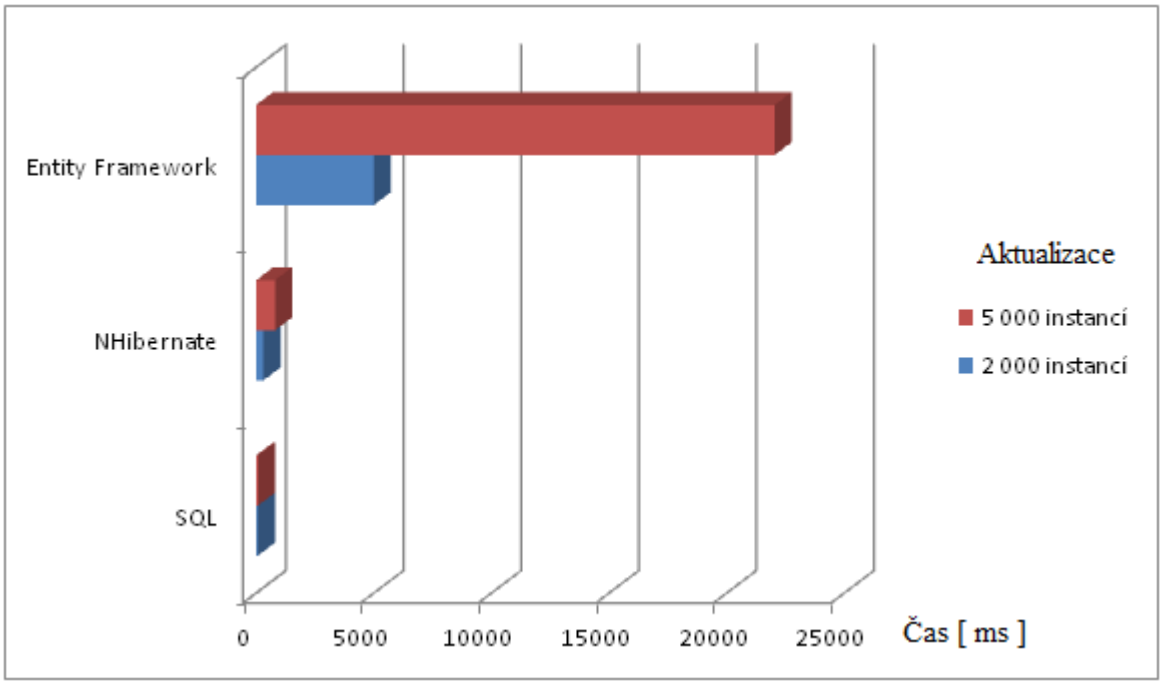
- Procesor - Intel Core2 Duo T5750, frekvence procesoru 2 GHz, frekvence sběrnice 667 MHz, L2 cache 2 MB
- Operační paměť - DDRII 3GB
- Pevný disk - Rozhraní SATA, 5400 ot. / min

Na grafu číslo 9.5 znázorňuji celkové výsledky pro operaci **vytváření.**



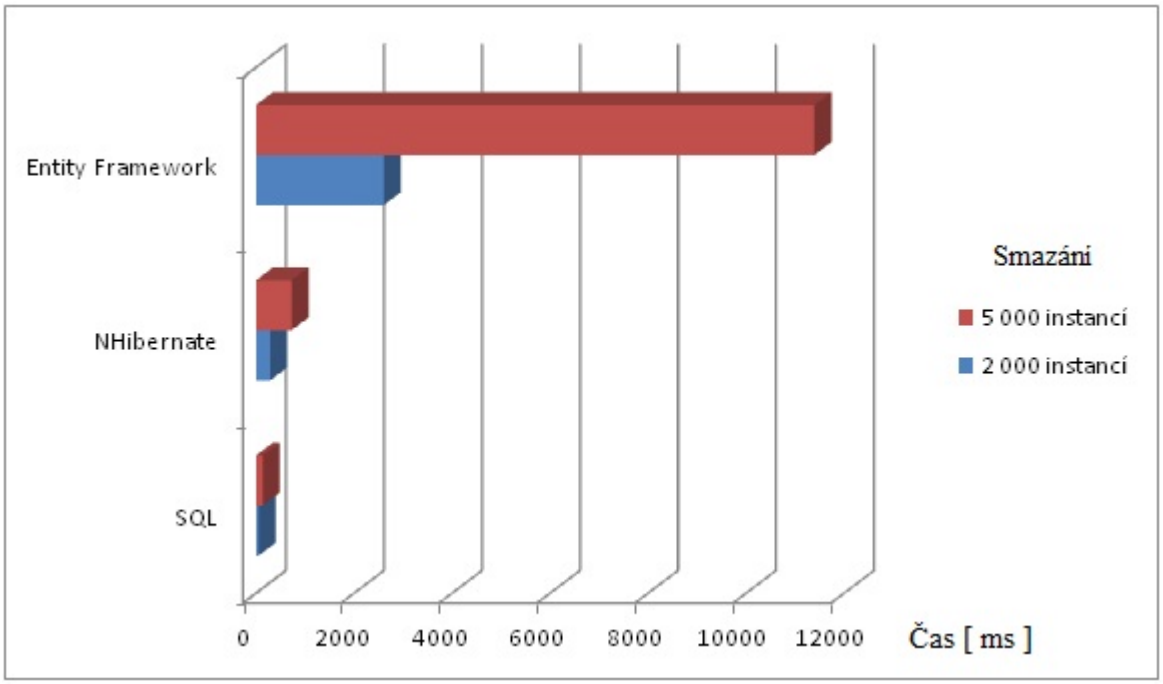
Graf 9.5 Celkové výsledky pro operaci vytváření

Na grafu číslo 9.6 znázorňuji celkové výsledky pro operaci **aktualizace.**



Graf číslo 9.6 Celkové výsledky pro operaci aktualizace

Na grafu 9.7 znázorňuji celkové výsledky pro operaci **smazání.**



Graf 9.7 Celkové výsledky pro operaci smazání.

9.3 Vyhodnocení dosažených výsledků

Entity Framework dosahoval nejhorších výsledků z hlediska času zpracování. Horší výsledky jsou zapříčiněny složitějším rozvrstvením rámce. Nástroj se snaží o poskytování konceptuální vrstvy. Z tohoto důvodu probíhá dvojitý překlad dotazu. Konceptuální vrstva tohoto rámce přináší výhody u systémů s potřebou vysokou mírou abstrakce. Z grafu 9.3 lze vyčíst, že s rostoucím počtem operací vzrůstá také čas na jednu operaci a proto je z hlediska datové propustnosti vhodnější na operace, jako jé dávkové zpracování dat, využít nástroj s nižší úrovní abstrakce. Pro IS Třanovice je plně dostačující. To dokazuje tabulka 9.5, v které lze vidět, že správce lokality je schopen vytvořit volné termíny na 3 měsíce dopředu během 13 sekund.

Kapitola IV. - Implementace

Poslední část bakalářské práce obsahuje výčet a stručný popis technologií, které jsem použil při implementaci informačního systému. Dále u každé technologie stručně popisuji, k čemu byla při implementaci použita.

10 POUŽITÉ TECHNOLOGIE

10.1 ASP.NET MVC 3 Framework

Pro implementaci webové části informační systému jsem zvolil tento Framework. V kapitole II. se tímto frameworkem zabývám detailně.

10.2 ASP.NET Web Forms

Pro implementaci testování ORM nástrojů Entity Framework a NHibernate jsem zvolil ASP.NET WebForms, které ulehčují práci při programování pomocí využívání ovládacích prvků (*Controls*).

10.3 SQL Server Express 2008

Microsoft SQL Server Express je relační databázový systém vhodný pro použití na desktopu, serveru i na webu. Je snadno použitelný, instalovatelný a dostupný zdarma včetně práva dalšího šíření s vaší aplikací. SQL Server Express 2008 jsem používal po celou dobu implementace jako databázový servr.

10.4 ADO.NET Entity Framework

ADO.NET Entity Framework je nástroj pro ORM mapování, tento mapovací nástroj jsem použil v informačním systému a dále pak v testovací části, kde jsem měřil propustnost dat tohoto nástroje. Tento nástroj se používá i na oficiálních stránkách www.asp.net, ze kterých jsem se hodně naučil především z videoukázek.

10.5 NHibernate

NHibernate je rovněž ORM rámec pro .NET. Pomocí XML popisu entit a vztahů NHibernate automaticky generuje SQL pro načítání a ukládání objektů.

10.6 JavaScript

Javascript je objektově orientovaný skriptovací jazyk, který běží na straně klienta a zpravidla je používaný pro ovládání interaktivních GUI prvků (tlačítka, formuláře atd.), tvorbu animací a efektů obrázků na WWW stránkách.

Já jsem javascript využil k získání dat z kalendáře, konkrétně když uživatel vybírá den, na který si chce zarezervovat danou sportovní lokalitu tak vyvolá javascript, který posílá data dále do aplikace na zpracování.

10.7 CSS

CSS, česky „ kaskádové styly", jsou jazykem pro formátování vzhledu HTML a XHTML dokumentů, který byl vytvořen za účelem oddělení formy dokumentu od jeho obsahu.

CSS jsou ve zde popisovaném systému uloženy ve zvláštních souborech v kořenovém adresáři ve složce Content. Při implementaci informačního systému jsem rozšířil tyto soubory o nastavení vzhledu těch částí systému, které jsem nově vytvořil. V případě nasazení systému pro jinou obec je třeba CSS přepsat pro dosažení požadovaného vzhledu.

10.8 jQuery a jQuery u.i.

jQuery je malá javascriptová knihovna, která klade důraz na interakci mezi JavaScriptem a HTML. Byla vydána Johnem Resigem v lednu 2006 na newyorském BarCampu. Z jQuery používám Ajaxovou komunikaci pro výpis termínu pro jednotlivé dny.

Dále používám v informačním systému DatePicker z jQuery, což je kalendář přes který provádí uživatel výběr data pro rezervaci.

ZÁVĚR

V rámci této bakalářské práce jsem navrhnul a implementoval informační systém pro obec Třanovice. Tento informační systém jsem implementoval v jednom z nejmodernějším frameworků pro webové informační systémy postaveném na návrhovém vzoru Model View Controller. Systém je snadno rozšiřitelný a po nasazení se systém může rozšířit o další funkcionalitu, například o možnost provedení fakturace pro uživatele využívající tento systém.

V rámci této práce jsem se také zabýval problematikou objektově-relačního mapování. Za účelem měření propustnosti dat mezi databází a aplikací jsem vytvořil sadu měření a naimplementoval jsem měřící nástroje pro NHibernate a Entity Framework. Nakonec jsem provedená měření vyhodnotil.

CD příloha pro tuto bakalářskou práci obsahuje veškeré zdrojové soubory informačního systému včetně zdrojových souborů měřících nástrojů.

LITERATURA

[1] ORACLE. *Your First Cup: An Introduction to the JavaTM EE Platform* [online]. Duben. 2012 [cit. 2012-07-25]. Dostupné z: http://docs.oracle.com/javaee/6/firstcup/doc/firstcup.pdf

[2] STUDENTSKÝ SERVR. *J2EE - Java 2 Enterprise Edition* [online]. [cit. 2012-07-25]. Dostupné z: http://artax.karlin.mff.cuni.cz/~ebik/nju/linuxem/J2EE/J2EE.html

[3] ORACLE. *J2EE - Documentation* [online]. duben. 2012 [cit. 2012-07-25]. Dostupné z: http://www.oracle.com/technetwork/java/javaee/documentation/index.html

[4] FREEMAN, Adam. *Pro ASP.NET MVC 3 framework*. 3rd. ed. New York: Apress, c2011, 824 s. The expert's Voice in.NET. ISBN 978-1-4302-3404-3.

[5] Objektově relační mapování. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-03]. Dostupné z: http://cs.wikipedia.org/wiki/Objektov%C4%9B_rela%C4%8Dn%C3%AD_mapov%C3%A1n%C3%AD

[6] FATRDLA, Pavel. *Porovnání technologií pro objektově relační mapování.* Diplomová práce, Vysoké učení technické v Brně, 2010

[7] JEŘÁBEK, Tomáš. *Implementace objektově-relačního mapování v datové vrstvě informačního systému pro platformu .NET.* Diplomová práce, VŠB - Technická univerzita Ostrava, 2011 Dostupné z: http://dspace.vsb.cz/handle/10084/87098

[8] MICROSOFT. *MSDN - Documentation* [online]. 2012 [cit. 2012-07-25]. Dostupné z: http://msdn.microsoft.com/en-us/data/aa937709.aspx

[9] NHIBERNATE FORGE. *NHibernate - Documentation* [online]. [cit. 2012-07-25]. Dostupné z: http://www.nhforge.org/doc/nh/en/index.html

PŘÍLOHY

Příloha A - Obsah doprovodného CD

1. Readme.txt - soubor s popisem datového CD.

2. Projekt IS_Třanovice - složka obsahující IS_Tranovice.sln, projekt byl vytvořen ve Visual Studiu 2010.

3. Databáze - Databáze k projektu IS_Tranovice, uložena pod názvem IS_Tranovice.mdf.

4. Text - bakalářská práce ve formátu PDF.